D 8

```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$

Sy
--
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI

LI
LI
LI
LI
LI
LI
LI

LI
LI
LI

LI
LI
LI
LI
LI
LI
```

```
**FILE**ID**OTSCVTDT                                        K 11




          OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  DDDDDDDD    TTTTTTTTTT
          OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  DDDDDDDD    TTTTTTTTTT
        OO      OO      TT      SS          CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT      SS          CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT      SS          CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT      SS          CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT        SSSSSS    CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT        SSSSSS    CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT              SS  CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT              SS  CC          VV      VV      TT      DD      DD      TT
        OO      OO      TT              SS  CC            VV  VV        TT      DD      DD      TT          ....
        OO      OO      TT              SS  CC            VV  VV        TT      DD      DD      TT          ....
          OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      DDDDDDDD        TT          ....
          OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      DDDDDDDD        TT          ....


        LL            IIIIII      SSSSSSSS
        LL            IIIIII      SSSSSSSS
        LL              II      SS
        LL              II      SS
        LL              II      SS
        LL              II      SS
        LL              II        SSSSSS
        LL              II        SSSSSS
        LL              II              SS
        LL              II              SS
        LL              II              SS
        LL              II              SS
        LLLLLLLLLL    IIIIII    SSSSSSSS
        LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000         .TITLE  OTS$$CVTDT
0000         .IDENT  /1-017/                  ; File: OTSCVTDT.MAR  Edit: LEB1017
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ;++
0000    ; FACILITY: Language-independent Support Library
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       A routine to convert an F or D-floating value to a string of
0000    ;       ASCII digits and an exponent.  It is meant to be used as
0000    ;       a base for floating point output conversion routines.
0000    ;
0000    ; ENVIRONMENT: User Mode, AST Reentrant
0000    ;
0000    ;--
0000    ; AUTHOR: Steven B. Lionel, CREATION DATE: 24-May-1979
0000    ;
```

```
0000           .SBTTL  Edit History
0000    ;
0000    ; 1-001 - Original.  Numeric conversion algorithm by Tryggve Fossum.
0000    ;         SBL 24-May-1979
0000    ; 1-002 - Make routine an _R8 so as to conform with OTS$$CVTRT.  SBL 3-Jul-1979
0000    ; 1-003 - Add extra longword to stack frame to prevent clobbering of
0000    ;         saved info.  SBL 8-Jul-1979
0000    ; 1-004 - Don't use R9 or R10 at all.  SBL 11-Jul-1979
0000    ; 1-005 - On right-rounding to zero, don't change the sign. SBL 16-Jul-1979
0000    ; 1-006 - Fix typo in stack frame setup.  SBL 23-July-1979
0000    ; 1-007 - Modify rounding algorithm so that if RT_RND would cause
0000    ;         rounding to the right of the number of significant digits,
0000    ;         the latter is used instead.  This is at the request of
0000    ;         BASIC - the situation can not occur in FORTRAN.  SBL 27-Jul-1979
0000    ; 1-008 - Clear 96 bits ahead of fraction instead of 63.  SBL 30-July-1979
0000    ; 1-009 - Speed improvements. Clear 64 bits ahead of fraction.  Use
0000    ;         register in inner convert loop.  SBL 21-Jan-1980
0000    ; 1-010 - Compute number of fraction longwords correctly at INIT_FRACT,
0000    ;         to assure accurate low-order digits.  JAW 21-Jul-1981
0000    ; 1-011 - Make sure all bits between significand and binary point are
0000    ;         cleared when value is an integer, to assure accurate low-order
0000    ;         digits.  JAW 26-Jul-1981
0000    ; 1-012 - If we find a reserved operand, return zero if it doesn't get
0000    ;         replaced by a non-reserved value.  SBL 29-Oct-81
0000    ; 1-013 - Add entry for F_floating.  SBL 29-Oct-1982
0000    ; 1-014 - Remove CVTFD instruction from OTS$$CVT_F_T_R8.  SBL 27-Apr-1983
0000    ; 1-015 - Fix bug introduced by 1-014.  SBL 17-May-1983
0000    ; 1-016 - Removed the CVTLP, CVTPS, and SKPC instructions to improve the
0000    ;         performance of this routine. Instead, EDIV instructions were
0000    ;         used. I also fixed a couple of comments.  JCW 31-OCT-1983
0000    ; 1-017 - Move tables after PSECT definition.  LEB 22-Mar-1984
0000
```

```
                               0000          .SBTTL  DECLARATIONS
                               0000   ;
                               0000   ; INCLUDE FILES:
                               0000   ;
                               0000
                               0000   ;
                               0000   ; EXTERNAL DECLARATIONS:
                               0000   ;
                               0000          .DSABL  GBL                              ; Prevent undeclared
                               0000                                                   ; symbols from being
                               0000                                                   ; automatically global.
                               0000
                               0000   ;
                               0000   ; MACROS:
                               0000   ;
                               0000
                               0000   ;
                               0000   ; EQUATED SYMBOLS:
                               0000   ;
                               0000
                               0000   ;
                               0000   ; PSECT DECLARATIONS:
                               0000   ;
                           00000000          .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
                               0000                          EXE, RD, NOWRT, LONG
                               0000
                               0000   ;
                               0000   ; OWN STORAGE:
                               0000   ;
                               0000   ; CONSTANTS
                               0000   ;
                               0000
                               0000   ASCII_ZEROES:
          30303030 30303030    0000          .QUAD    ^X3030303030303030            ; 8 copies of the character 0
                               0008
3430 3330 3230 3130 3030       0008   TABLE:  .WORD    ^X3030, ^X3130, ^X3230, ^X3330, ^X3430
3930 3830 3730 3630 3530       0012          .WORD    ^X3530, ^X3630, ^X3730, ^X3830, ^X3930
3431 3331 3231 3131 3031       001C          .WORD    ^X3031, ^X3131, ^X3231, ^X3331, ^X3431
3931 3831 3731 3631 3531       0026          .WORD    ^X3531, ^X3631, ^X3731, ^X3831, ^X3931
3432 3332 3232 3132 3032       0030          .WORD    ^X3032, ^X3132, ^X3232, ^X3332, ^X3432
3932 3832 3732 3632 3532       003A          .WORD    ^X3532, ^X3632, ^X3732, ^X3832, ^X3932
3433 3333 3233 3133 3033       0044          .WORD    ^X3033, ^X3133, ^X3233, ^X3333, ^X3433
3933 3833 3733 3633 3533       004E          .WORD    ^X3533, ^X3633, ^X3733, ^X3833, ^X3933
3434 3334 3234 3134 3034       0058          .WORD    ^X3034, ^X3134, ^X3234, ^X3334, ^X3434
3934 3834 3734 3634 3534       0062          .WORD    ^X3534, ^X3634, ^X3734, ^X3834, ^X3934
3435 3335 3235 3135 3035       006C          .WORD    ^X3035, ^X3135, ^X3235, ^X3335, ^X3435
3935 3835 3735 3635 3535       0076          .WORD    ^X3535, ^X3635, ^X3735, ^X3835, ^X3935
3436 3336 3236 3136 3036       0080          .WORD    ^X3036, ^X3136, ^X3236, ^X3336, ^X3436
3936 3836 3736 3636 3536       008A          .WORD    ^X3536, ^X3636, ^X3736, ^X3836, ^X3936
3437 3337 3237 3137 3037       0094          .WORD    ^X3037, ^X3137, ^X3237, ^X3337, ^X3437
3937 3837 3737 3637 3537       009E          .WORD    ^X3537, ^X3637, ^X3737, ^X3837, ^X3937
3438 3338 3238 3138 3038       00A8          .WORD    ^X3038, ^X3138, ^X3238, ^X3338, ^X3438
3938 3838 3738 3638 3538       00B2          .WORD    ^X3538, ^X3638, ^X3738, ^X3838, ^X3938
3439 3339 3239 3139 3039       00BC          .WORD    ^X3039, ^X3139, ^X3239, ^X3339, ^X3439
3939 3839 3739 3639 3539       00C6          .WORD    ^X3539, ^X3639, ^X3739, ^X3839, ^X3939
                               00D0
                               00D0   ; Stack frame offsets from R7
```

```
                 00D0   134 ;; Common frame for kernel convert routines
        FFFFFFF8 00D0   135         PACKED = -8                             ; Temp for packed representation
        FFFFFFF4 00D0   136         FLAGS = PACKED - 4                      ; Flags for outer and inner routines
        FFFFFFF0 00D0   137         SIG_DIGITS = FLAGS - 4                  ; Significant digits
        FFFFFFEC 00D0   138         STRING_ADDR = SIG_DIGITS - 4            ; Address of temp string
        FFFFFFE8 00D0   139         SIGN = STRING_ADDR - 4                  ; Sign
        FFFFFFE4 00D0   140         DEC_EXP = SIGN - 4                      ; Decimal exponent
        FFFFFFE0 00D0   141         OFFSET = DEC_EXP - 4                    ; Offset
        FFFFFFDC 00D0   142         RT_RND = OFFSET - 4                     ; Right round point
        FFFFFFDC 00D0   143         COMMON_FRAME = RT_RND                   ; Common frame size
                 00D0   144
                 00D0   145 ;+
                 00D0   146 ; Inner routine frame pointed to by R8 during conversion
                 00D0   147 ;-
        FFFFFFF0 00D0   148         INT_HI = -16                            ; Highest integer part
        FFFFFFE4 00D0   149         BIN_PT = INT_HI - 12                    ; Binary point
        FFFFFFC8 00D0   150         FRACT_LIM = BIN_PT - 28                 ; Lowest fraction bits
        FFFFFFB4 00D0   151         DIGITS = FRACT_LIM - 20                 ; Digits radix 10**9
        FFFFFFB0 00D0   152         BIN_EXP = DIGITS - 4                    ; Saved binary exponent
        FFFFFFAC 00D0   153         LONG_COUNT = BIN_EXP - 4                ; Longword count
        FFFFFFA8 00D0   154         TEMP = LONG_COUNT - 4                   ; Temporary
        FFFFFFA8 00D0   155         LOCAL_FRAME = TEMP                      ; Local frame size
                 00D0   156
                 00D0   157
```

```
          00D0    .SBTTL  OTS$$CVT_D_T  - Convert D floating to text
          00D0 ;++
          00D0 ; FUNCTIONAL DESCRIPTION:
          00D0 ;
          00D0 ;       This routine converts a D-floating point value to a string
          00D0 ;       of ASCII digits.  It is intended to form the base of a
          00D0 ;       language's floating point output conversion routine.
          00D0 ;
          00D0 ;       OTS$$CVT_F_T_R8 converts F_floating.
          00D0 ;
          00D0 ; CALLING SEQUENCE:
          00D0 ;
          00D0 ;       MOVAB   common_frame, R1          ; See common_frame definition above
          00D0 ;       MOVL    string_length, STRING_LEN(R1)
          00D0 ;       MOVL    string_address, STRING_ADDR(R1)
          00D0 ;       MOVL    sig_digits, SIG_DIGITS(R1)
          00D0 ;       MOVL    user_flags, FLAGS(R1)
          00D0 ;       MOVL    rt_round, RT_RND(R1)      ;  Optional
          00D0 ;       MOVAB   value, R0
          00D0 ;       JSB     OTS$$CVT_D_T_R8 or OTS$$CVT_F_T_R8
          00D0 ;       ; outputs are:
          00D0 ;       ;       R1 = unchanged
          00D0 ;       ;       OFFSET(R1) - offset
          00D0 ;       ;       DEC_EXP(R1) - decimal exponent
          00D0 ;       ;       SIGN(R1) - sign
          00D0 ;
          00D0 ; INPUT PARAMETERS:
          00D0 ;
          00D0 ;       VALUE                                     ; F or D-floating value to be converted
          00D0 ;       SIG_DIGITS(R1)                            ; Number of significant digits to
          00D0 ;                                                 ; generate.  If neither V_TRUNCATE
          00D0 ;                                                 ; or V_ROUND_RIGHT is set, the
          00D0 ;                                                 ; value will be rounded to this
          00D0 ;                                                 ; many digits.
          00D0 ;       FLAGS(R1)                                 ; Caller supplied flags:
00000018  00D0             V_TRUNCATE = 24                       ; Truncate, don't round.
00000019  00D0             V_ROUND_RIGHT = 25                    ; Round "rt_round" digits to
          00D0                                                   ; right of decimal point.
          00D0 ;       RT_RND(R1)                                ; Number of places to the right
          00D0 ;                                                 ; of the decimal point to round
          00D0 ;                                                 ; after.  Ignored if V_ROUND_RIGHT
          00D0 ;                                                 ; is clear.  The rounding takes
          00D0 ;                                                 ; place after the specified number
          00D0 ;                                                 ; of significant digits if that
          00D0 ;                                                 ; would be farther to the left.
          00D0 ;
          00D0 ; IMPLICIT INPUTS:
          00D0 ;
          00D0 ;       NONE
          00D0 ;
          00D0 ; OUTPUT PARAMETERS:
          00D0 ;
          00D0 ;       out_string                                ; String with result.  It will
          00D0 ;                                                 ; Not have valid digits after the
          00D0 ;                                                 ; requested number of significant
          00D0 ;                                                 ; digits.
          00D0 ;                                                 ; The length MUST be at least:
```

```
00D0                                                   ; (9*INT((sig_digits+17)/9))+2
00D0      ;        offset                              ; The offset into out_string at
00D0                                                   ; which the first significant digit
00D0                                                   ; may be found.  It is guaranteed
00D0                                                   ; to be between 0 and 9.
00D0      ;        exponent                            ; The signed decimal exponent of
00D0                                                   ; the value, assuming a radix point
00D0                                                   ; immediately to the left of the
00D0                                                   ; most significant digit.
00D0      ;        sign                                ; -1 if the value is negative
00D0                                                   ; 0 if the value is zero
00D0                                                   ; 1 if the value is positive
00D0      ;
00D0      ; IMPLICIT OUTPUTS:
00D0      ;
00D0      ;        NONE
00D0      ;
00D0      ; SIDE EFFECTS:
00D0      ;
00D0      ;        Alters registers R0 through R8.
00D0      ;
00D0      ;        SS$_ROPRAND      - If the value is a reserved operand
00D0      ;        SS$_ACCVIO       , or other nasty errors if the length of
00D0      ;                           out_string is not enough (see formula above).
00D0      ;                           This routine does not check the length, it
00D0      ;                           is up to the caller to insure the correct
00D0      ;                           length is present.
00D0      ;
00D0      ;--
00D0
```

```
                                00D0        .SBTTL  OTS$$CVT_D_T_R8
                                00D0
                                00D0  ;+
                                00D0  ; JSB entry point
                                00D0  ;-
                                00D0
                                00D0  OTS$$CVT_F_T_R8::
                  57  51  D0    00D0        MOVL    R1, R7                  ; Use R7 as common frame pointer
                      51  D4    00D3        CLRL    R1                      ; Clear high part of value
                  50  60  50    00D5        MOVF    (R0), R0                ; Fetch and test for zero
                      06  11    00D8        BRB     COMMON_FD               ; Join common code
                                00DA
                                00DA  OTS$$CVT_D_T_R8::
                  57  51  D0    00DA        MOVL    R1, R7                  ; Use R7 as common frame pointer
                  50  60  70    00DD        MOVD    (R0), R0                ; Fetch and test for zero
                                00E0                                        ; and for reserved operand
                                00E0  COMMON_FD:
                      0E  14    00E0        BGTR    VAL_POS                 ; Value is positive
                      06  19    00E2        BLSS    VAL_NEG                 ; Value is negative
                      7D  10    00E4        BSBB    ZERO                    ; Value is zero
                  51  57  D0    00E6        MOVL    R7, R1                  ; Restore R1
                          05    00E9        RSB                             ; Return to caller
                                00EA
                                00EA  VAL_NEG:
              E8 A7   01  CE    00EA        MNEGL   #1, SIGN(R7)            ; Set negative sign
                      04  11    00EE        BRB     EXTRACT                 ; Continue
                                00F0  VAL_POS:
              E8 A7   01  D0    00F0        MOVL    #1, SIGN(R7)            ; Set positive sign
                                00F4
                                00F4  EXTRACT:
                  58  5E  D0    00F4        MOVL    SP, R8                  ; R8 points to local frame
              5E   A8 AE  9E    00F7        MOVAB   LOCAL_FRAME(SP), SP     ; Set up local frame
              54   E4 A8  9E    00FB        MOVAB   BIN_PT(R8), R4          ; R4 points to binary point
        52    50   08 07  EF    00FF        EXTZV   #7, #8, R0, R2          ; Extract exponent
                      5D  13    0104        BEQL    ZERO                    ; Still reserved operand; give up
              50   50 10  9C    0106        ROTL    #16, R0, R0             ; Make into proper fraction
              51   51 10  9C    010A        ROTL    #16, R1, R1
        52    00000080 8F C2    010E        SUBL2   #128, R2                ; Remove bias
                      05  18    0115        BGEQ    10$                     ; If value is less than 1,
                   F8 A4  7C    0117        CLRQ    -8(R4)                  ;  clear some fraction bits
                                011A                                        ;  in case value is < 2**-64.
                      02  11    011A        BRB     20$
                      64  7C    011C  10$:  CLRQ    (R4)                    ; If value is greater than 1,
                                011E                                        ;  clear some integer bits
                                011E                                        ;  in case value is >= 2**88.
      F5 A4   20   52 00  F0    011E  20$:  INSV    #0, R2, #32, -11(R4)    ; Create fixed point binary
      F9 A4   20   52 51  F0    0124        INSV    R1, R2, #32, -7(R4)     ; value with enough surrounding
         50   00800000 8F C8    012A        BISL2   #^X800000, R0
      FD A4   18   52 50  F0    0131        INSV    R0, R2, #24, -3(R4)     ; zeroes as "guard digits".
         64   20   52 00  F0    0137        INSV    #0, R2, #32, (R4)
      04 A4   20   52 00  F0    013C        INSV    #0, R2, #32, +4(R4)
              B0 A8   52  D0    0142        MOVL    R2, BIN_EXP(R8)         ; Save binary exponent
                      2D  15    0146        BLEQ    FRACT_ONLY              ; If less than 1...
              56   B4 A8  9E    0148        MOVAB   DIGITS(R8), R6          ; R6 points to scratch area
                      57  DD    014C        PUSHL   R7                      ; Save R7 so we can use as temp
        55    52   FB 8F  78    014E        ASHL    #-5, R2, R5             ; How many integer longwords?
              AC A8   55  D0    0153        MOVL    R5, LONG_COUNT(R8)
```

```
                      07   15 0157    304             BLEQ    ONE_LONG                      ; 1 longword
                03    55   91 0159    305             CMPB    R5, #3
                      29   19 015C    306             BLSS    INT_LOOP                      ; 2 or 3 longwords
                      1C   11 015E    307             BRB     FOUR_LONG                     ; Four longwords
                              0160    308
                              0160    309 ONE_LONG:
                    007B   31 0160    310             BRW     INT_NEXT
                              0163    311
                              0163    312 ZERO:
                51 EC A7   D0 0163    313             MOVL    STRING_ADDR(R7), R1           ; Get string address
 61     F0 A7   30  6E 00  2C 0167    314             MOVC5   #0, (SP), #^A/0/, SIG_DIGITS(R7), (R1) ; Zero fill string
                   E0 A7   7C 016E    315             CLRQ    OFFSET(R7)                    ; Clear offset and exponent
                   E8 A7   D4 0171    316             CLRL    SIGN(R7)                      ; Zero has sign of zero
                           05 0174    317             RSB                                   ; Return to caller
                              0175    318
                              0175    319 FRACT_ONLY:
              AC A8   01   CE 0175    320             MNEGL   #1, LONG_COUNT(R8)            ; To note that there is no integer part
                    00E0   31 0179    321             BRW     FORMAT                        ; Go directly to formatter
                              017C    322
```

```
                                   017C       .SBTTL  Numeric conversion routines
                                   017C
                                   017C  ;+
                                   017C  ; This is the portion which converts the integer part of the value
                                   017C  ; to 1-5 longwords of radix 10**9.  This is done by repeated division
                                   017C  ; by 10**9.
                                   017C  ;-
                                   017C  FOUR_LONG:
                          51   D4  017C          CLRL    R1                           ; High part of dividend
                 54    F0 A8   DE  017E          MOVAL   INT_HI(R8), R4               ; Use R4 as address pointer
                    50    64   D0  0182          MOVL    (R4),R0                      ; Low part of dividend
                          14   11  0185          BRB     INT_DIV
                                   0187  INT_LOOP:
               54    E4 A845   DE  0187          MOVAL   BIN_PT(R8)[R5], R4           ; Get address pointer
                       04 A4   D5  018C          TSTL    4(R4)                        ; Are we missing some bits?
                          07   13  018F          BEQL    10$                          ; No if zero
                          55   D6  0191          INCL    R5                           ; Back up one longword
                       AC A8   D6  0193          INCL    LONG_COUNT(R8)               ; Bump longword counter
                          EF   11  0196          BRB     INT_LOOP                     ; And try again
                    50    64   7D  0198  10$:    MOVQ    (R4), R0                     ; Get first quadword of dividend
                                   019B  INT_DIV:
 51   64   50   3B9ACA00 8F    7B  019B          EDIV    #^D1000000000,R0,(R4),R1
                       04 A4   D4  01A4          CLRL    4(R4)                        ; Since this is really a
                                   01A7                                               ; quadword quotient, zero the
                                   01A7                                               ; higher longword.
                    57    55   D0  01A7          MOVL    R5, R7                       ; R7 is inner loop counter
      53   51   1DCD6500 8F    C3  01AA  30$:    SUBL3   #^D500000000,R1,R3           ; Is this dividend too large ?
                          15   19  01B2          BLSS    40$                          ; Skip adjustment if not
                    52    74   D0  01B4          MOVL    -(R4), R2                    ; Low part of dividend
 51   64   52   3B9ACA00 8F    7B  01B7          EDIV    #^D1000000000,R2,(R4),R1
                                   01C0                                               ; Divide by 10**9
           64   80000000 8F    C8  01C0          BISL    #^X80000000,(R4)             ; Set high bit
                          0C   11  01C7          BRB     60$
                    50    74   D0  01C9  40$:    MOVL    -(R4), R0                    ; Get low part of dividend
 51   64   50   3B9ACA00 8F    7B  01CC          EDIV    #^D1000000000,R0,(R4),R1
                                   01D5                                               ; Divide and store result in (R4)
                       D2 57   F5  01D5  60$:    SOBGTR  R7, 30$                      ; Loop back
                    86    51   D0  01D8          MOVL    R1, (R6)+                    ; Store result on stack
                       A9 55   F5  01DB          SOBGTR  R5, INT_LOOP                 ; loop back if not done
                                   01DE  INT_NEXT:
                 52    AC A8   D0  01DE          MOVL    LONG_COUNT(R8), R2
                 50    E4 A8   7D  01E2          MOVQ    BIN_PT(R8),R0                ; Low part of next dividend
 86    04 A6   50   3B9ACA00 8F    7B  01E6      EDIV    #^D1000000000,R0,4(R6),(R6)+
                          02   13  01F0          BEQL    10$                          ; Branch if high longword is 0
                          52   D6  01F2          INCL    R2                           ; Convert one more longword
                     B4 A842   D5  01F4  10$:    TSTL    DIGITS(R8)[R2]               ; Find first non-zero longword
                          04   12  01F8          BNEQ    20$                          ; Found.  Go format them.
                          52   D7  01FA          DECL    R2                           ; Not found.  Try next one.
                          F6   11  01FC          BRB     10$
                    AC A8 52   D0  01FE  20$:    MOVL    R2, LONG_COUNT(R8)           ; Save longword count
                       57 8E   D0  0202          MOVL    (SP)+, R7                    ; Restore R7 from where saved
                          55   11  0205          BRB     FORMAT
                                   0207
                                   0207  ;+
                                   0207  ; This routine initializes the pointer for getting fraction digits.
                                   0207  ; The number of fraction longwords is calculated and is stored in
                                   0207  ; LONG_COUNT(R8) for future calls.
```

```
                                       0207     ;-
                                       0207     INIT_FRACT:
    50   B0 A8   00000057 8F      C3   0207                 SUBL3   #<56+32-1>, BIN_EXP(R8), R0
            AC A8      50 20      C7   0210                 DIVL3   #32, R0, LONG_COUNT(R8)
                                       0215
                                       0215     ;+
                                       0215     ; This routine gets the next nine fraction digits.  It is smart
                                       0215     ; enough not to do EMULs on zero values.
                                       0215     ;-
                                       0215     GET_FRACT:
                          51      D4   0215                 CLRL    R1                              ; Result is initially zero
                   52   AC A8     D0   0217                 MOVL    LONG_COUNT(R8), R2              ; Get number of fraction longwords
                          1A      18   021B                 BGEQ    30$                             ; If not negative, return
              53   E4 A842        DE   021D     5$:         MOVAL   BIN_PT(R8)[R2], R3              ; Get address of lowest longword
                       50 63      D0   0222     10$:        MOVL    (R3), R0                        ; Get the longword
                          11      15   0225                 BLEQ    40$                             ; Beware of overflow on EMUL
    50    51   50  3B9ACA00 8F    7A   0227                 EMUL    #^D1000000000, R0, R1, R0
                       83 50      D0   0230                 MOVL    R0, (R3)+                       ; Store result
                          52      D6   0233                 INCL    R2                              ; 1 less longword
                          EB      19   0235                 BLSS    10$                             ; Loop back if more
                          05           0237     30$:        RSB
                          18      13   0238     40$:        BEQL    60$                             ; Don't multiply a zero
    50    51   50  3B9ACA00 8F    7A   023A                 EMUL    #^D1000000000, R0, R1, R0
               51  3B9ACA00 8F    C0   0243                 ADDL2   #^D1000000000, R1               ; To prevent overflow
                       83 50      D0   024A                 MOVL    R0, (R3)+                       ; Store result
                          52      D6   024D                 INCL    R2                              ; 1 less longword
                          D1      19   024F                 BLSS    10$                             ; Loop back if more
                          05           0251                 RSB
                       83 51      D0   0252     60$:        MOVL    R1, (R3)+                       ; Store current product
                          51      D4   0255                 CLRL    R1
                          52      D6   0257                 INCL    R2                              ; 1 less longword
                          C7      19   0259                 BLSS    10$                             ; Loop back if more
                          05           025B                 RSB
```

```
                              025C           .SBTTL  Character formatting routines
                              025C  ;+
                              025C  ; After all the integer portion of the value has been converted to
                              025C  ; longwords and stored, the integer part is then converted to
                              025C  ; characters and the fraction part, if any, is converted.
                              025C  ;-
                              025C  FORMAT:
         55    EC A7      D0  025C          MOVL    STRING_ADDR(R7), R5          ; Get string address
            85    30      90  0260          MOVB    #^A/0/, (R5)+                ; Set first character to '0'
      56    F0 A7  01     C1  0263          ADDL3   #1, SIG_DIGITS(R7), R6       ; Generate at least one extra digit
         50    AC A8      D0  0268          MOVL    LONG_COUNT(R8), R0           ; How many integer longwords?
                  03      18  026C          BGEQ    1$
                00F7      31  026E          BRW     NO_INT                       ; If none, skip this part
            55    09      C0  0271  1$:     ADDL2   #9, R5                       ; R5 will store least signif digit
                              0274                                               ;  (lsd) in the high order byte.
            53    55      D0  0274          MOVL    R5, R3                       ; save the old address
      F7 A5   FD85 CF     7D  0277          MOVQ    ASCII_ZEROES, -9(R5)         ; Initialize the string to contain 30's
                              027D                                               ;  the 9th byte will be filled below
         51    B4 A840    D0  027D          MOVL    DIGITS(R8)[R0], R1           ; R1/R2 must be a quadword for
                  52      D4  0282          CLRL    R2                           ;  the EDIV
54   51   51  00000064 8F     7B  0284          EDIV    #100, R1, R1, R4             ; extract two lsd
                  31      13  028D          BEQL    60$
          75   FD74 CF44  B0  028F          MOVW    TABLE[R4], -(R5)             ; load correct char rep of the 2 digits
54   51   51  00000064 8F     7B  0295          EDIV    #100, R1, R1, R4             ; extract two lsd
                  20      13  029E          BEQL    60$
          75   FD63 CF44  B0  02A0          MOVW    TABLE[R4], -(R5)             ; load correct char rep of the 2 digits
54   51   51  00000064 8F     7B  02A6          EDIV    #100, R1, R1, R4             ; extract two lsd
                  0F      13  02AF          BEQL    60$
          75   FD52 CF44  B0  02B1          MOVW    TABLE[R4], -(R5)             ; load correct char rep of the 2 digits
54   51   51  00000064 8F     7B  02B7          EDIV    #100, R1, R1, R4             ; extract two lsd
          75   FD43 CF44  B0  02C0  60$:    MOVW    TABLE[R4], -(R5)             ; load correct char rep of the 2 digits
          75    51    30      81  02C6          ADDB3   #^A/0/, R1, -(R5)            ; character rep needed for last number
                              02CA  ;
                              02CA  ; Numbers are stored as characters as follows: low order byte is the most
                              02CA  ; significant digit (character), while the high order byte is the least signif
                              02CA  ; digit (character). The storage took place from the high oder digit to the
                              02CA  ; low order digit. Since we used an EDIV by 100, 0,1, or 2 zeroes may be
                              02CA  ; located at (R5). R0 is to contain the number of nonzero digits (not char 30)
                              02CA  ; between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one
                              02CA  ; zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130
                              02CA  ; while 102 --> 3230313030.
                              02CA  ;
       50    53    55     C3  02CA          SUBL3   R5, R3, R0
                   51     D5  02CE          TSTL    R1
                   0A     12  02D0          BNEQ    98$
                   50     D7  02D2          DECL    R0                           ; At least on leading 30.
          30    01 A5     91  02D4          CMPB    1(R5), #^A/0/                ; there still could be 1 more 0
                   02     12  02D8          BNEQ    98$                          ; 102 --> 3230313030 by the above
                              02DA                                               ;  we've already seen rightmost 30
                   50     D7  02DA          DECL    R0                           ;  if there is another, subt 1.
                              02DC                                               ;  There can be no more consect 0's
    E0 A7    0A    50     C3  02DC  98$:    SUBL3   R0, #10, OFFSET(R7)
    51    AC A8    09     C5  02E1          MULL3   #9, LONG_COUNT(R8), R1       ; Calculate exponent
    E4 A7    51    50     C1  02E6          ADDL3   R0, R1, DEC_EXP(R7)          ; Store exponent
             55    53     D0  02EB          MOVL    R3, R5                       ; Move string pointer up by 9
             56    50     C2  02EE          SUBL2   R0, R6                       ; Decrease # of digits left to produce
                              02F1  OUT_LOOP:
```

```
                          72  15  02F1          BLEQ    OUT_ROUND               ; Done if no more sig. digits
                       AC A8  D7  02F3          DECL    LONG_COUNT(R8)          ; Decrement longword count
                   50  AC A8  D0  02F6          MOVL    LONG_COUNT(R8), R0
                          03  18  02FA          BGEQ    1$
                        00F8  31  02FC          BRW     OUT_FRACT               ; Do fraction part if time
                   55     09  C0  02FF  1$:     ADDL2   #9, R5                  ; R5 will store least signif digit
                                  0302                                          ;  (lsd) in the high order byte.
                   53     55  D0  0302          MOVL    R5, R3                  ; save the old address
             F7 A5   FCF7 CF  7D  0305          MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                  030B                                          ;  the 9th byte will be filled below
                51     B4 A840  D0  030B        MOVL    DIGITS(R8)[R0], R1      ;
                   0A     51  D1  0310          CMPL    R1, #^X000000A          ; if R1 < 10 you may skip the EDIV
                          44  19  0313          BLSS    70$                     ;
                          52  D4  0315          CLRL    R2                      ; R1/R2 must be a quadword for the EDIV
  54    51    51  00000064 8F  7B  0317         EDIV    #100, R1, R1, R4        ; extract two lsd
                          31  13  0320          BEQL    60$
                75   FCE1 CF44  B0  0322        MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54    51    51  00000064 8F  7B  0328         EDIV    #100, R1, R1, R4        ; extract two lsd
                          20  13  0331          BEQL    60$
                75   FCD0 CF44  B0  0333        MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54    51    51  00000064 8F  7B  0339         EDIV    #100, R1, R1, R4        ; extract two lsd
                          0F  13  0342          BEQL    60$
                75   FCBF CF44  B0  0344        MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54    51    51  00000064 8F  7B  034A         EDIV    #100, R1, R1, R4        ; extract two lsd
                75   FCB0 CF44  B0  0353  60$:  MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
             75    51     30  81  0359  70$:    ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                   55     53  D0  035D          MOVL    R3, R5                  ; Move string pointer up by 9
                   56     09  C2  0360          SUBL2   #9, R6                  ; Adjust # of sig. digits
                          8C  11  0363          BRB     OUT_LOOP
                                  0365
                                  0365  OUT_ROUND:
                                  0365  ;       BRB     ROUND
                        0157  31  0365          BRW     ROUND
                                  0368
                                  0368  ;+
                                  0368  ; This code is executed if the value is less than 1.
                                  0368  ;-
                                  0368  NO_INT:
                        FE9C  30  0368          BSBW    INIT_FRACT              ; Initialize the pointers
                                  036B                                          ; and get first 9 digits.
                       E4 A7  D4  036B          CLRL    DEC_EXP(R7)             ; Calculate exponent
                 E4 A7    09  C2  036E  10$:    SUBL2   #9, DEC_EXP(R7)         ; Its 9 smaller now
                          51  D5  0372          TSTL    R1                      ; Are digits zero?
                          05  12  0374          BNEQ    20$
                        FE9C  30  0376          BSBW    GET_FRACT               ; Get next 9 digits
                          F3  11  0379          BRB     10$                     ; And try again
                   55     09  C0  037B  20$:    ADDL2   #9, R5                  ; R5 will store least signif digit
                                  037E                                          ;  (lsd) in the high order byte.
                   53     55  D0  037E          MOVL    R5, R3                  ; save the old address
             F7 A5   FC7B CF  7D  0381          MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                  0387                                          ;  the 9th byte will be filled below
                   0A     51  D1  0387          CMPL    R1, #^X000000A          ; if R1 < 10 you may skip the EDIV
                          44  19  038A          BLSS    70$                     ;
                          52  D4  038C          CLRL    R2                      ; R1/R2 must be a quadword for the EDIV
  54    51    51  00000064 8F  7B  038E         EDIV    #100, R1, R1, R4        ; extract two lsd
                          31  13  0397          BEQL    60$
                75   FC6A CF44  B0  0399        MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  039F   529           EDIV    #100, R1, R1, R4          ; extract two lsd
                          20   13  03A8   530           BEQL    60$
                 75  FC59 CF44  B0  03AA   531           MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  03B0   532           EDIV    #100, R1, R1, R4          ; extract two lsd
                          0F   13  03B9   533           BEQL    60$
                 75  FC48 CF44  B0  03BB   534           MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  03C1   535           EDIV    #100, R1, R1, R4          ; extract two lsd
                 75  FC39 CF44  B0  03CA   536 60$:      MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
             75   51   30      81  03D0   537 70$:      ADDB3   #^A/0/, R1, -(R5)         ; character rep needed for last number
                                   03D4   538 ;
                                   03D4   539 ; Numbers are stored as characters as follows: low order byte is the most
                                   03D4   540 ; significant digit (character), while the high order byte is the least signif
                                   03D4   541 ; digit (character). The storaye took place from the high oder digit to the
                                   03D4   542 ; low order digit. Since we used an EDIV by 100, 0,1, or 2 zeroes may be
                                   03D4   543 ; located at (R5). R0 is to contain the number of nonzero digits (not char 30)
                                   03D4   544 ; between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one
                                   03D4   545 ; zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130
                                   03D4   546 ; while 102 --> 3230313030.
                                   03D4   547 ;
             50   53   55      C3  03D4   548           SUBL3   R5, R3, R0
                       51      D5  03D8   549           TSTL    R1
                       0A      12  03DA   550           BNEQ    98$
                       50      D7  03DC   551           DECL    R0                        ; At least on leading 30.
             30   01   A5      91  03DE   552           CMPB    1(R5), #^A/0/             ; there still could be 1 more 0
                       02      12  03E2   553           BNEQ    98$                       ; 102 --> 3230313030 by the above
                                   03E4   554                                             ;  we've already seen rightmost 30
                       50      D7  03E4   555           DECL    R0                        ;  if there is another, subt 1.
                                   03E6   556                                             ;  There can be no more consect 0's
         E0 A7    0A   50      C3  03E6   557 98$:      SUBL3   R0, #10, OFFSET(R7)
              E4  A7   50      C0  03EB   558           ADDL2   R0, DEC_EXP(R7)           ; Calculate exponent
                  55   53      D0  03EF   559           MOVL    R3, R5                    ; Move string pointer up by 9
                  56   50      C2  03F2   560           SUBL2   R0, R6                    ; Adjust # of sig. digits
                       62      11  03F5   561           BRB     FRACT_LOOP                ; Get 9 more
                                   03F7   562
                                   03F7   563 ;+
                                   03F7   564 ; This code starts the fraction portion if the integer portion exists.
                                   03F7   565 ;-
                                   03F7   566 OUT_FRACT:
                     FE0D      30  03F7   567           BSBW    INIT_FRACT                ; Initialize and get 9 digits
                  55   09      C0  03FA   568           ADDL2   #9, R5                    ; R5 will store least signif digit
                                   03FD   569                                             ;  (lsd) in the high order byte.
                  53   55      D0  03FD   570           MOVL    R5, R3                    ; save the old address
         F7 A5    FBFC CF      7D  0400   571           MOVQ    ASCII_ZEROES, -9(R5)      ; Initialize the string to contain 30's
                                   0406   572                                             ;  the 9th byte will be filled below
                  0A   51      D1  0406   573           CMPL    R1, #^X000000A            ; if R1 < 10 you may skip the EDIV
                       44      19  0409   574           BLSS    70$                       ;
                       52      D4  040B   575           CLRL    R2                        ; R1/R2 must be a quadword for the EDIV
  54   51   51   00000064 8F   7B  040D   576           EDIV    #100, R1, R1, R4          ; extract two lsd
                          31   13  0416   577           BEQL    60$
                 75  FBEB CF44  B0  0418   578           MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  041E   579           EDIV    #100, R1, R1, R4          ; extract two lsd
                          20   13  0427   580           BEQL    60$
                 75  FBDA CF44  B0  0429   581           MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  042F   582           EDIV    #100, R1, R1, R4          ; extract two lsd
                          0F   13  0438   583           BEQL    60$
                 75  FBC9 CF44  B0  043A   584           MOVW    TABLE[R4], -(R5)          ; load correct char rep of the 2 digits
  54   51   51   00000064 8F   7B  0440   585           EDIV    #100, R1, R1, R4          ; extract two lsd
```

```
               75  FBBA CF44  B0  0449   586 60$:    MOVW    TABLE[R4], -(R5)           ; load correct char rep of the 2 digits
                 75   51   30  81  044F   587 70$:    ADDB3   #^A/0/, R1, -(R5)          ; character rep needed for last number
                      55   53  D0  0453   588         MOVL    R3, R5                     ; Move string pointer up by 9
                      56   09  C2  0456   589         SUBL2   #9, R6                     ; Adjust # of sig. digits
                               0459   590 FRACT_LOOP:
                           64  15  0459   591         BLEQ    ROUND                      ; If not, finish
                         FDB7  30  045B   592         BSBW    GET_FRACT                  ; Get 9 more digits
                      55   09  C0  045E   593         ADDL2   #9, R5                     ; R5 will store least signif digit
                                   0461   594                                            ;  (lsd) in the high order byte.
                      53   55  D0  0461   595         MOVL    R5, R3                     ; save the old address
           F7 A5   FB98 CF  7D  0464   596         MOVQ    ASCII_ZEROES, -9(R5)       ; Initialize the string to contain 30's
                                   046A   597                                            ;  the 9th byte will be filled below
                      0A   51  D1  046A   598         CMPL    R1, #^X000000A             ; if R1 < 10 you may skip the EDIV
                           44  19  046D   599         BLSS    70$                        ;
                           52  D4  046F   600         CLRL    R2                         ; R1/R2 must be a quadword for the EDIV
 54    51    51    00000064 8F  7B  0471   601         EDIV    #100, R1, R1, R4           ; extract two lsd
                           31  13  047A   602         BEQL    60$
               75  FB87 CF44  B0  047C   603         MOVW    TABLE[R4], -(R5)           ; load correct char rep of the 2 digits
 54    51    51    00000064 8F  7B  0482   604         EDIV    #100, R1, R1, R4           ; extract two lsd
                           20  13  048B   605         BEQL    60$
               75  FB76 CF44  B0  048D   606         MOVW    TABLE[R4], -(R5)           ; load correct char rep of the 2 digits
 54    51    51    00000064 8F  7B  0493   607         EDIV    #100, R1, R1, R4           ; extract two lsd
                           0F  13  049C   608         BEQL    60$
               75  FB65 CF44  B0  049E   609         MOVW    TABLE[R4], -(R5)           ; load correct char rep of the 2 digits
 54    51    51    00000064 8F  7B  04A4   610         EDIV    #100, R1, R1, R4           ; extract two lsd
               75  FB56 CF44  B0  04AD   611 60$:    MOVW    TABLE[R4], -(R5)           ; load correct char rep of the 2 digits
                 75   51   30  81  04B3   612 70$:    ADDB3   #^A/0/, R1, -(R5)          ; character rep needed for last number
                      55   53  D0  04B7   613         MOVL    R3, R5                     ; Move string pointer up by 9
                      56   09  C2  04BA   614         SUBL2   #9, R6                     ; Adjust # of sig. digits
                           9A  11  04BD   615         BRB     FRACT_LOOP                 ; Loop back for more
                                   04BF   616
                                   04BF   617 ;+
                                   04BF   618 ; This routine rounds the value to the given number of significant
                                   04BF   619 ; digits, unless flag V_TRUNCATE is on.  If so, the value is truncated
                                   04BF   620 ; at the next digit.
                                   04BF   621 ;-
                                   04BF   622 ROUND:
                           56  D7  04BF   623         DECL    R6
                      55   56  C0  04C1   624         ADDL2   R6, R5                     ; Find least significant + 1
            41 F4 A7   18  E0  04C4   625         BBS     #V_TRUNCATE, FLAGS(R7), FINIS   ; Truncate if desired
            17 F4 A7   19  E1  04C9   626         BBC     #V_ROUND_RIGHT, FLAGS(R7), 5$   ; Round to right of dec pt?
     50    DC A7   E4 A7  C1  04CE   627         ADDL3   DEC_EXP(R7), RT_RND(R7), R0     ; Yes, find it
                           34  19  04D4   628         BLSS    FINIS                      ; Done if rounds to zero
               F0 A7   50  D1  04D6   629         CMPL    R0, SIG_DIGITS(R7)         ; Round to right of # sig digits?
                           09  18  04DA   630         BGEQ    5$                         ; Yes, round to significant digits
               50   E0 A7  C0  04DC   631         ADDL2   OFFSET(R7), R0             ; Finish calculation
        55    EC A7   50  C1  04E0   632         ADDL3   R0, STRING_ADDR(R7), R5    ; Get rounding character address
                      35   65  91  04E5   633 5$:     CMPB    (R5), #^A/5/               ; Round?
                      20   19  04E8   634         BLSS    FINIS                      ; No, just finish
                      50   55  D0  04EA   635         MOVL    R5, R0                     ; Save position
                      39   70  91  04ED   636 10$:    CMPB    -(R0), #^A/9/              ; If this is a 9...
                      05   19  04F0   637         BLSS    20$
                      60   30  90  04F2   638         MOVB    #^A/0/, (R0)               ; Then it becomes a zero
                      F6   11  04F5   639         BRB     10$                        ; And we continue
                      60   96  04F7   640 20$:    INCB    (R0)                       ; Else this is last carry
               50   EC A7  C2  04F9   641         SUBL2   STRING_ADDR(R7), R0        ; Do we need to change offset
               E0 A7   50  D1  04FD   642         CMPL    R0, OFFSET(R7)             ; and exponent?
```

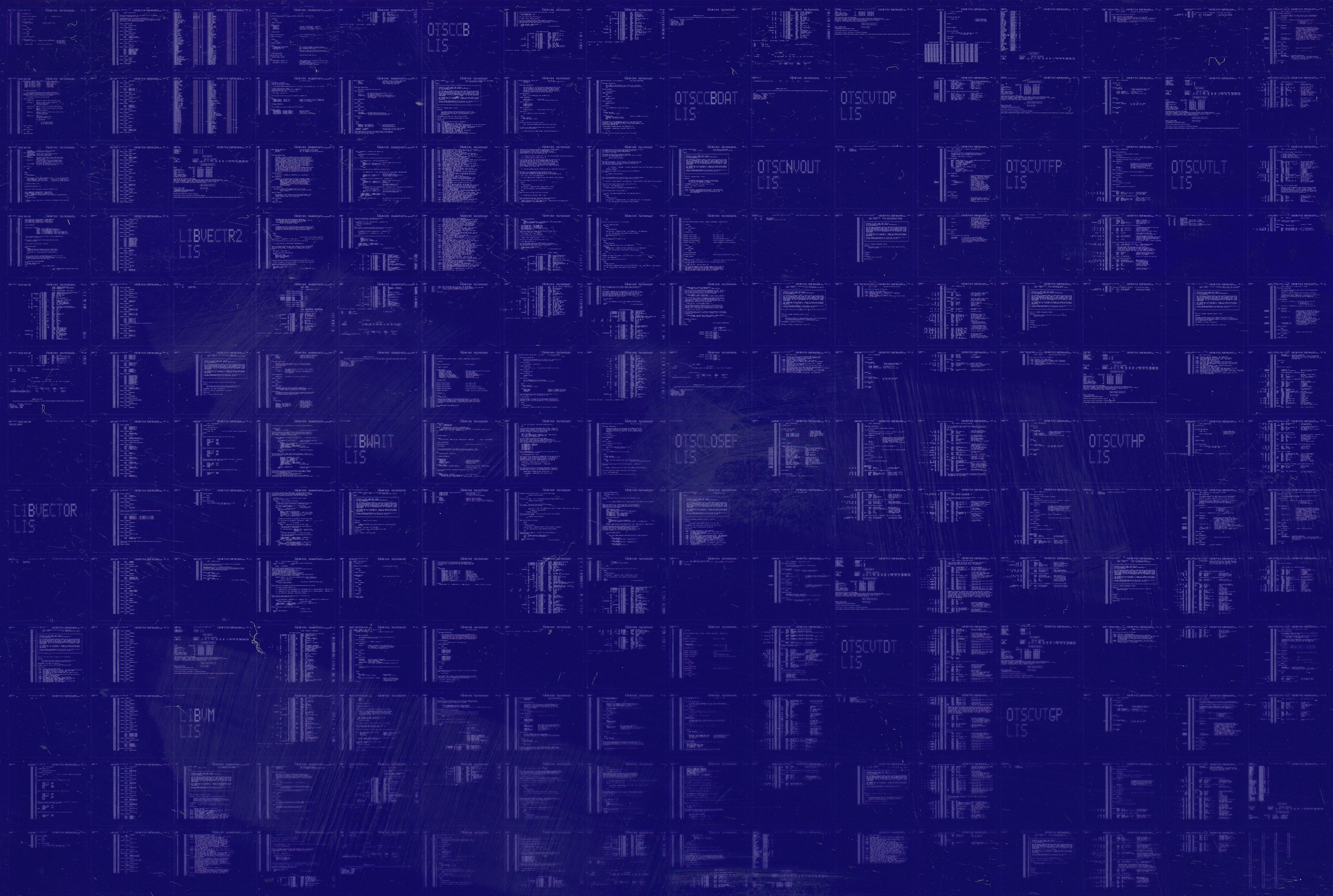
```
                07  18  0501          BGEQ    FINIS                   ; No
         E0 A7  50  D0  0503          MOVL    R0, OFFSET(R7)          ; Yes, set new offset
            E4 A7   D6  0507          INCL    DEC_EXP(R7)             ; Set new exponent
                        050A
                        050A    ;+
                        050A    ; All done.
                        050A    ;-
                        050A    FINIS:
5E   FFFFFFA8  8F   C2  050A          SUBL2   #LOCAL_FRAME, SP        ; Restore stack pointer
           51  57   D0  0511          MOVL    R7, R1                  ; Restore common frame pointer
                    05  0514          RSB                             ; Return to caller
                        0515
                        0515          .END
```

```
ASCII_ZEROES         00000000 R       01
BIN_EXP            = FFFFFFB0
BIN_PT             = FFFFFFE4
COMMON_FD            000000EC R       01
DEC_EXP            = FFFFFFE4
DIGITS             = FFFFFFB4
EXTRACT              000000F4 R       01
FINIS                0000050A R       01
FLAGS              = FFFFFFF4
FORMAT               0000025C R       01
FOUR_LONG            0000017C R       01
FRACT_LIM          = FFFFFFC8
FRACT_LOOP           00000459 R       01
FRACT_ONLY           00000175 R       01
GET_FRACT            00000215 R       01
INIT_FRACT           00000207 R       01
INT_DIV              0000019B R       01
INT_HI             = FFFFFFF0
INT_LOOP             00000187 R       01
INT_NEXT             000001DE R       01
LOCAL_FRAME        = FFFFFFA8
LONG_COUNT         = FFFFFFAC
NO_INT               00000368 R       01
OFFSET             = FFFFFFE0
ONE_LONG             00000160 R       01
OTS$$CVT_D_T_R8      000000DA RG      01
OTS$$CVT_F_T_R8      000000D0 RG      01
OUT_FRACT            000003F7 R       01
OUT_LOOP             000002F1 R       01
OUT_ROUND            00000365 R       01
PACKED             = FFFFFFF8
ROUND                000004BF R       01
RT_RND             = FFFFFFDC
SIGN               = FFFFFFE8
SIG_DIGITS         = FFFFFFF0
STRING_ADDR        = FFFFFFEC
TABLE                00000008 R       01
TEMP               = FFFFFFA8
VAL_NEG              000000EA R       01
VAL_POS              000000F0 R       01
V_ROUND_RIGHT      = 00000019
V_TRUNCATE         = 00000018
ZERO                 00000163 R       01


                                     +-----------------+
                                     ! Psect synopsis  !
                                     +-----------------+

PSECT name                Allocation          PSECT No.  Attributes
----------                ----------          ---------  ----------
.  ABS  .                 00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
_OTS$CODE                 00000515  (  1301.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG
```

```
                                            +-------------------------+
                                            ! Performance indicators !
                                            +-------------------------+

Phase                        Page faults     CPU Time        Elapsed Time
-----                        -----------     --------        ------------
Initialization                        33     00:00:00.02     00:00:01.62
Command processing                   135     00:00:00.30     00:00:02.78
Pass 1                                94     00:00:01.28     00:00:07.73
Symbol table sort                      0     00:00:00.03     00:00:00.03
Pass 2                               130     00:00:00.86     00:00:04.28
Symbol table output                    5     00:00:00.03     00:00:00.34
Psect synopsis output                  2     00:00:00.02     00:00:00.02
Cross-reference output                 0     00:00:00.00     00:00:00.00
Assembler run totals                 401     00:00:02.54     00:00:16.80

The working set limit was 1200 pages.
13641 bytes (27 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 44 non-local and 31 local symbols.
655 source lines were read in Pass 1, producing 10 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

                                           +--------------------------+
                                           ! Macro library statistics !
                                           +--------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:OTSCVTDT/OBJ=OBJ$:OTSCVTDT MSRC$:OTSCVTDT/UPDATE=(ENH$:OTSCVTDT)
```

0211 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



OTSCCB
LIS

OTSCCBDAT
LIS

OTSCVTDP
LIS

OTSCNVOUT
LIS

OTSCVTFP
LIS

OTSCVTLT
LIS

LIBVECTR2
LIS

LIBWAIT
LIS

OTSCLOSEF
LIS

OTSCVTHP
LIS

LIBVECTOR
LIS

OTSCVTDT
LIS

LIBVM
LIS

OTSCVTGP
LIS